Assignment-4

```
#include <GL/freeglut.h>
#include <GL/gl.h>
#include<stdio.h>
double x1,x2,y1,y2;
//struct pixel to store color in rgb format
typedef struct pixel
{
        float r,g,b;
}pixel;
pixel f_clr,b_clr;//fill color and boundary color
void init()
{
        glClearColor(1.0, 1.0, 1.0, 0.0);//to set background color
   glClear(GL_COLOR_BUFFER_BIT);//to apply back ground color to screen
   glColor3f(0.0, 0.0, 0.0);//to set color of object to be shown on screen
   gluOrtho2D(0,500,0,500);//to set 2D projection rectangle
}
//dda line drawing algorithm
void dda(double x1,double y1,double x2,double y2)
{
        double dx,dy,steps;
        float xi,yi;
        dx=x2-x1;
        dy=y2-y1;
        steps=abs(dx)>abs(dy)?abs(dx):abs(dy);
        xi=dx/(float)steps;
        yi=dy/(float)steps;
        int i;
        glBegin(GL_POINTS);
        glColor3f(0.0, 0.0, 0.0);
```

```
        glVertex2d(x1,y1);
        for(i=0;i<steps;i++)
        {
                x1+=xi;
                y1+=yi;
                glVertex2d(x1,y1);//function to print one pixel
        }
        glEnd();
        glFlush();
}
//boundary fill algorithm
void boundary_fill(int x,int y)
{
        pixel c;
        glReadPixels(x,y,1.0,1.0,GL_RGB,GL_FLOAT,&c);
        if((c.r!=b_clr.r||c.g!=b_clr.g||c.b!=b_clr.b)&&(c.r!=f_clr.r||c.g!=f_clr.g||c.b!=f_clr.b))
        {
                glBegin(GL_POINTS);
                glColor3f(f_clr.r,f_clr.g,f_clr.b);
                glVertex2i(x,y);
                glEnd();
                glFlush();
                boundary_fill(x-1,y);
                boundary_fill(x+1,y);
                boundary_fill(x,y-1);
                boundary_fill(x,y+1);
        }
        glFlush();
}
int ch=1,a,b,c,d;
//function defined for mouse handling
```

```
void mymouse(int btn,int state,int x,int y)
{
	if(btn==GLUT_LEFT_BUTTON&&state==GLUT_DOWN)//if left click
	{
		switch(ch)
		{
			//for selecting vertex in polygon to fill
			case 1:
				a=x;
				b=500-y;
				ch=2;
				break;
			//for selecting color in color pallette
			case 2:
				c=x;
				d=500-y;
				glReadPixels(c,d,1.0,1.0,GL_RGB,GL_FLOAT,&f_clr);
				boundary_fill(a,b);
				ch=1;
				break;
		}
	}
}
void renderFunction()
{
   int xmax,ymax,i;
   glClear(GL_COLOR_BUFFER_BIT);
   //making color palette
   dda(100,400,400,400);
   dda(100,450,400,450);
   dda(100,400,100,450);
```

```
dda(400,400,400,450);
for(i=1;i<=4;i++)
{
        dda(100+i*60,400,100+i*60,450);
}
//coloring color palette
f_clr.r=1.0f;
f_clr.g=0.0f;
f_clr.b=0.0f;
boundary_fill(130,425);
f_clr.r=0.0f;
f_clr.g=1.0f;
f_clr.b=0.0f;
boundary_fill(190,425);
f_clr.r=1.0f;
f_clr.g=1.0f;
f_clr.b=0.0f;
boundary_fill(250,425);
f_clr.r=0.0f;
f_clr.g=0.0f;
f_clr.b=1.0f;
boundary_fill(310,425);
f_clr.r=0.0f;
f_clr.g=1.0f;
f_clr.b=1.0f;
boundary_fill(370,425);
//making diagram
dda(100,100,200,100);
dda(200,100,170,75);
dda(170,75,330,75);
dda(330,75,300,100);
```

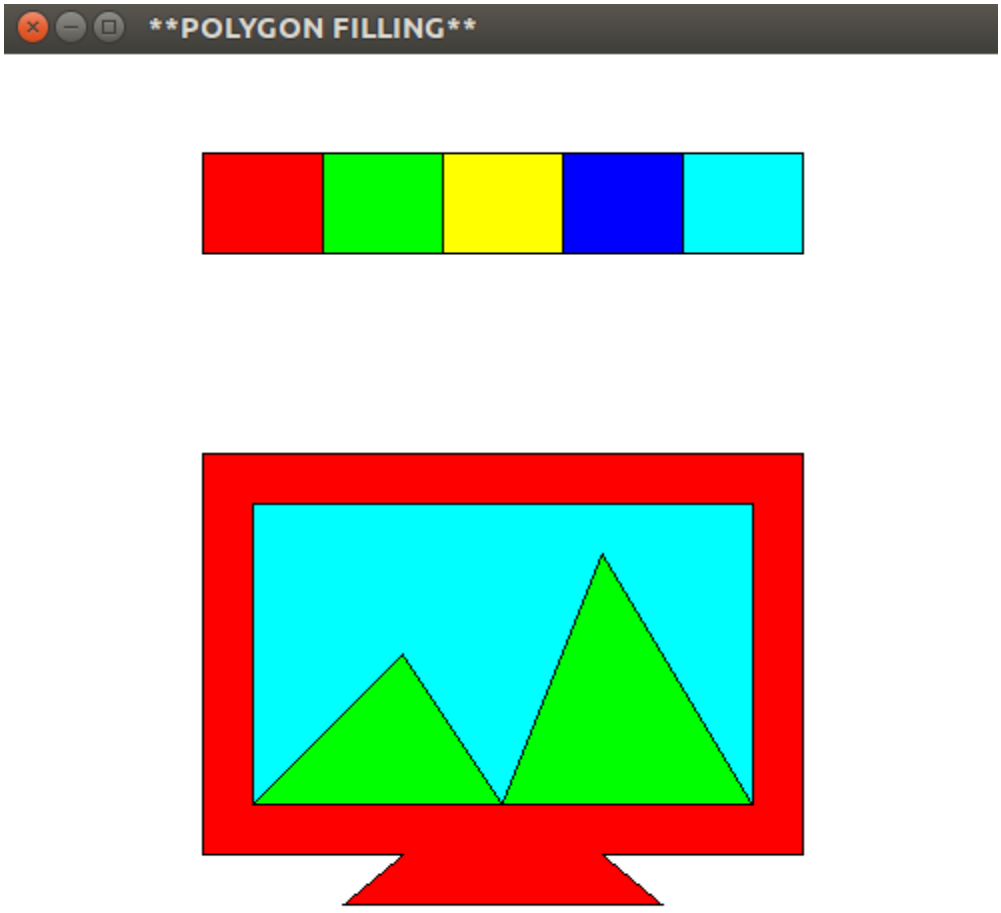
```
    dda(300,100,400,100);
    dda(100,100,100,300);
    dda(100,300,400,300);
    dda(400,300,400,100);
    dda(125,125,375,125);
    dda(125,125,125,275);
    dda(125,275,375,275);
    dda(375,275,375,125);
    dda(125,125,200,200);
    dda(200,200,250,125);
    dda(250,125,300,250);
    dda(300,250,375,125);
    //setting boundary color
    b_clr.r=0.0f;
    b_clr.g=0.0f;
    b_clr.b=0.0f;
    glFlush();
}
int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE);
    glutInitWindowSize(500,500);
    glutInitWindowPosition(100,100);
    glutCreateWindow("**POLYGON FILLING**");
    init();
    glutDisplayFunc(renderFunction);
    glutMouseFunc(mymouse);
    glutMainLoop();
    return 0;
}
```

```
#include <GL/freeglut.h>
#include <GL/gl.h>
#include<stdio.h>
double x1,x2,y1,y2;
//struct pixel to store color in rgb format
typedef struct pixel
{
        float r,g,b;
}pixel;
pixel f_clr,b_clr;//fill color and boundary color
void init()
{
   glClearColor(1.0, 1.0, 1.0, 0.0);//to set background color
   glClear(GL_COLOR_BUFFER_BIT);//to apply back ground color to screen
   glColor3f(0.0, 0.0, 0.0);//to set color of object to be shown on screen
   gluOrtho2D(0,500,0,500);//to set 2D projection rectangle
}
void dda(double x1,double y1,double x2,double y2)
{
        double dx,dy,steps;
        float xi,yi;
        dx=x2-x1;
        dy=y2-y1;
        steps=abs(dx)>abs(dy)?abs(dx):abs(dy);
        xi=dx/(float)steps;
        yi=dy/(float)steps;
        int i;
        glBegin(GL_POINTS);
        glColor3f(0.0, 0.0, 0.0);
        glVertex2d(x1,y1);
```

```
for(i=0;i<steps;i++)
        {
                x1+=xi;
                y1+=yi;
                glVertex2d(x1,y1);
        }
        glEnd();
        glFlush();
}
void boundary_fill(int x,int y)
{
        pixel c;
        glReadPixels(x,y,1.0,1.0,GL_RGB,GL_FLOAT,&c);//function to read color
        if((c.r!=b_clr.r||c.g!=b_clr.g||c.b!=b_clr.b)&&(c.r!=f_clr.r||c.g!=f_clr.g||c.b!=f_clr.b))
        {
                glBegin(GL_POINTS);
                glColor3f(f_clr.r,f_clr.g,f_clr.b);
                glVertex2i(x,y);
                glEnd();
                glFlush();
                boundary_fill(x-1,y);
                boundary_fill(x+1,y);
                boundary_fill(x,y-1);
                boundary_fill(x,y+1);
        }
        glFlush();
}
void renderFunction()
{
        int i;
   glClear(GL_COLOR_BUFFER_BIT);
```

```
    dda(100,100,200,100);
    dda(200,100,170,75);
    dda(170,75,330,75);
    dda(330,75,300,100);
    dda(300,100,400,100);
    dda(100,100,100,300);
    dda(100,300,400,300);
    dda(400,300,400,100);
    dda(125,125,375,125);
    dda(125,125,125,275);
    dda(125,275,375,275);
    dda(375,275,375,125);
    dda(125,125,200,200);
    dda(200,200,250,125);
    dda(250,125,300,250);
    dda(300,250,375,125);
    b_clr.r=0.0f;
    b_clr.g=0.0f;
    b_clr.b=0.0f;
    f_clr.r=1.0f;
    f_clr.g=0.0f;
    f_clr.b=0.0f;
    boundary_fill(120,120);
    f_clr.r=0.0f;
    f_clr.g=1.0f;
    f_clr.b=0.0f;
    boundary_fill(135,130);
    boundary_fill(360,130);
    glFlush();
}
```

```
int main(int argc, char** argv)
{
   glutInit(&argc, argv);
   glutInitDisplayMode(GLUT_SINGLE);
   glutInitWindowSize(500,500);
   glutInitWindowPosition(100,100);
   glutCreateWindow("OpenGL - First window demo");
   init();
   glutDisplayFunc(renderFunction);
   glutMainLoop();
   return 0;
}
```


POLYGON FILLING

```
#include<stdio.h>
#include<GL/gl.h>
#include<GL/glu.h>
#include<GL/glut.h>
#include<math.h>
/*draw chess pattern rotate it and fill it with different colours*/
typedef struct pixel
{
GLubyte r,g,b;
}pixel;
pixel f_color,b_color;
float mat1[20][3];
float ans1[20][3];
float trans1[3][3];
int ch=1;
void initial_co()
{
int i,y,x;
y=90;
//horizontal lines
for(i=0;i<10;i+=2)
{
mat1[i][0]=90;
mat1[i][1]=y;
mat1[i][2]=1;
mat1[i+1][0]=210;
mat1[i+1][1]=y;
mat1[i+1][2]=1;
y+=30;
}
x=90;
```

```
for(i;i<20;i+=2)
{
mat1[i][0]=x;
mat1[i][1]=90;
mat1[i][2]=1;
//second point
mat1[i+1][0]=x;
mat1[i+1][1]=210;
mat1[i+1][2]=1;
x+=30;
}
}
void rotate_fig()
{
int i,j,k;
float theta;
theta=45*3.14/180;

/*----------------translation to origin ---------------------------------*/
for(i=0;i<3;i++)
{
for(j=0;j<3;j++)
{
if(i==j)
trans1[i][j]=1;
else
trans1[i][j]=0;
}
}
trans1[2][0]=trans1[2][1]=-150;
for(i=0;i<20;i++)
```

```
{
for(j=0;j<3;j++)
{
ans1[i][j]=0;
for(k=0;k<3;k++)
ans1[i][j]+=mat1[i][k]*trans1[k][j];
}
}
/*----------------------rotation at origin-------------------------------*/
for(i=0;i<3;i++)
{
for(j=0;j<3;j++)
{
if(i==j)
trans1[i][j]=1;
else
trans1[i][j]=0;
}
}
trans1[0][0]=trans1[1][1]=cos(theta);
trans1[0][1]=sin(theta);
trans1[1][0]=-sin(theta);
/*
trans1=  cos sin 0
-sin cos 0
0    0  1
*/

for(i=0;i<20;i++)
{
for(j=0;j<3;j++)
```

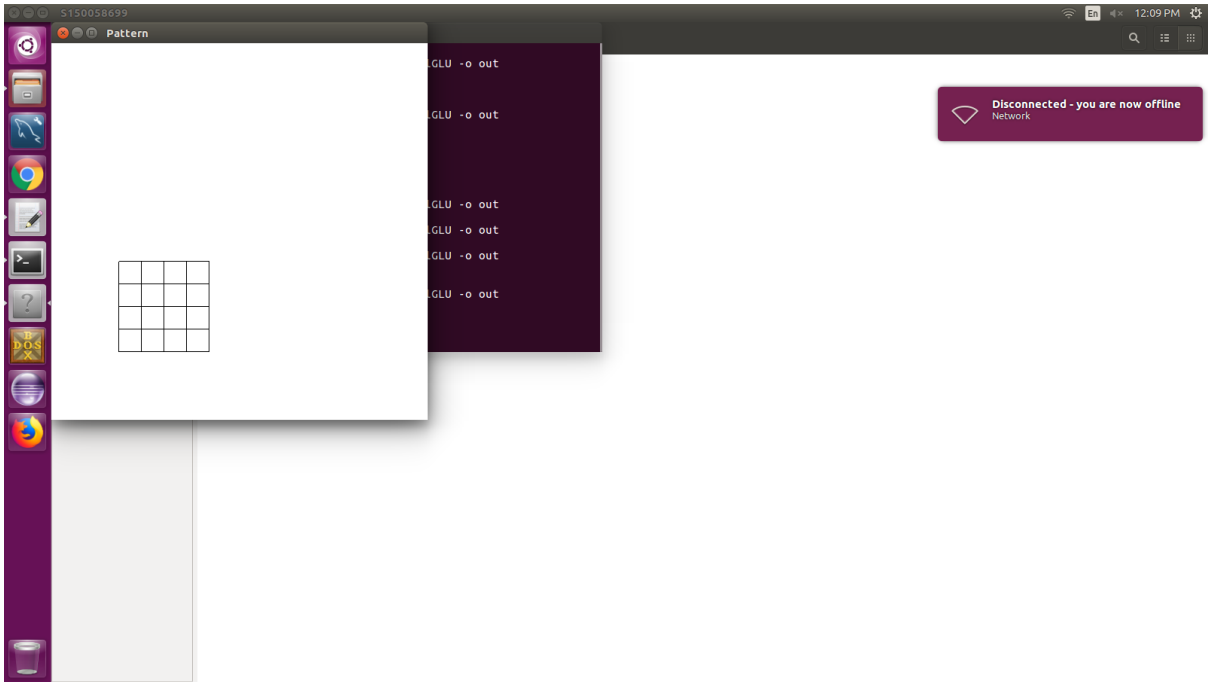
```
{
mat1[i][j]=0;
for(k=0;k<3;k++)
mat1[i][j]+=ans1[i][k]*trans1[k][j];
}
}
/*----------------------translation back-----------------------------*/
for(i=0;i<3;i++)
{
for(j=0;j<3;j++)
{
if(i==j)
trans1[i][j]=1;
else
trans1[i][j]=0;
}
}
trans1[2][0]=trans1[2][1]=150;
for(i=0;i<20;i++)
{
for(j=0;j<3;j++)
{
ans1[i][j]=0;
for(k=0;k<3;k++)
ans1[i][j]+=mat1[i][k]*trans1[k][j];
}
}
}
void boundary_fill(int x,int y)
{
pixel c;
```

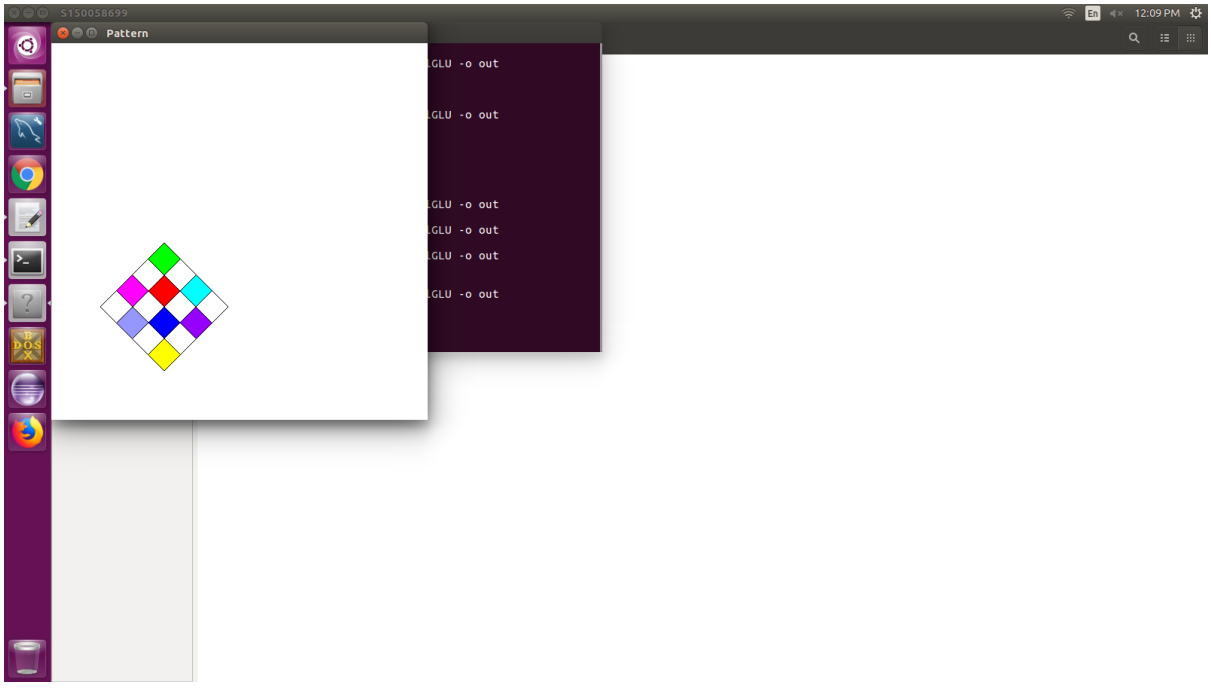
```
glReadPixels(x,y,1,1,GL_RGB,GL_UNSIGNED_BYTE,&c);//values are put into c
if((c.r!=b_color.r || c.g!=b_color.g || c.b!=b_color.b )&&(c.r!=f_color.r || c.g!=f_color.g || c.b!=f_color.b ))
{
glColor3ub(f_color.r,f_color.g,f_color.b);//set fill color for pixel
glBegin(GL_POINTS);
glVertex2d(x,y);//put pixel
glEnd();
glFlush();
boundary_fill(x+1,y);//right pixel
boundary_fill(x-1,y);//left pixel
boundary_fill(x,y+1);//upper pixel
boundary_fill(x,y-1);//lower pixel
}
}
void before()
{
int i;
initial_co();
glBegin(GL_LINES);//draws the new figure
for(i=0;i<20;i+=2)
{
glVertex2f(mat1[i][0],mat1[i][1]);
glVertex2f(mat1[i+1][0],mat1[i+1][1]);
}
glEnd();
glFlush();
}
void figure()
{
glClear(GL_COLOR_BUFFER_BIT);
```

```
int i;
float factor=30*cos(45*3.14/180);
rotate_fig();//rotates the figure about the middle point (150,150)
glBegin(GL_LINES);//draws the new figure
for(i=0;i<20;i+=2)
{
glVertex2f(ans1[i][0],ans1[i][1]);
glVertex2f(ans1[i+1][0],ans1[i+1][1]);
}
glEnd();
glFlush();
boundary_fill(150,150+factor);
f_color.r=0;
f_color.g=255;
f_color.b=0;
boundary_fill(150,150+3*factor);
f_color.r=0;
f_color.g=0;
f_color.b=255;
boundary_fill(150,150-factor);
f_color.r=255;
f_color.g=255;
f_color.b=0;
boundary_fill(150,150-3*factor);
f_color.r=0;
f_color.g=255;
f_color.b=255;
boundary_fill(150+2*factor,150+factor);
f_color.r=255;
f_color.g=0;
f_color.b=255;
```

```
boundary_fill(150-2*factor,150+factor);
f_color.r=150;
f_color.g=0;
f_color.b=255;
boundary_fill(150+2*factor,150-factor);
f_color.r=150;
f_color.g=150;
f_color.b=255;
boundary_fill(150-2*factor,150-factor);
}
void mouse_click(int btn,int state,int x,int y)
{
if(btn==GLUT_LEFT_BUTTON && state==GLUT_DOWN)
{
switch(ch)
{
case 1:
before();//initial figure
ch=2;
break;
case 2:
figure();//after transformation
ch=3;
break;
case 3:
break;
}
}
}
void init_func()//empty function doesnt do anything
{
```

```
glFlush();
}
void Init()
{
glClearColor(1.0,1.0,1.0,0.0);//sets the background colour
glClear(GL_COLOR_BUFFER_BIT);
glColor3f(0.0,0.0,0.0);//sets the drawing colour
gluOrtho2D(0,500,0,500);//sets the co ordinates
}
int main(int argc,char **argv)
{
b_color.r=b_color.g=b_color.b=0;
f_color.r=255;
f_color.g=0;
f_color.b=0;
glutInit(&argc,argv);//initializing the library
glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);//setting the display mode
glutInitWindowPosition(0,0);//position of the window
glutInitWindowSize(500,500);//size of the window
glutCreateWindow("Pattern");//name of the window
Init();//initializes the background colour and co ordinates
glutDisplayFunc(init_func);//displays the function
glutMouseFunc(mouse_click);//to display before and after figures
glutMainLoop();//keeps the program open until closed
return 0;
}
```


S150058699
12:09 PM
Pattern
.GLU -o out
.GLU -o out
.GLU -o out
.GLU -o out
.GLU -o out
.GLU -o out
Disconnected - you are now offline
Network


S150058699
12:09 PM
Pattern
.GLU -o out
.GLU -o out
.GLU -o out
.GLU -o out
.GLU -o out
.GLU -o out